SiCortex and Lustre

- Overview
  - The SiCortex system
- Lustre in the SiCortex system
  - Application data
  - Root filesystem
- The SiCortex LND
  - Interconnect/rdma
  - Implementation strategy
  - Special capabilities
  - Current status
- Summary
- Questions

SC5832

5832	Gigaflops
7776	Gigabytes ECC memory
972	6-core 64-bit nodes
2916	2 GByte/s fabric links
500	GByte/s bisection bandwidth
1	µs MPI latency
270	GByte/s I/O bandwidth
108	8-lane PCI Express
18	KW (208v 3Ø 60A)
1	Cabinet

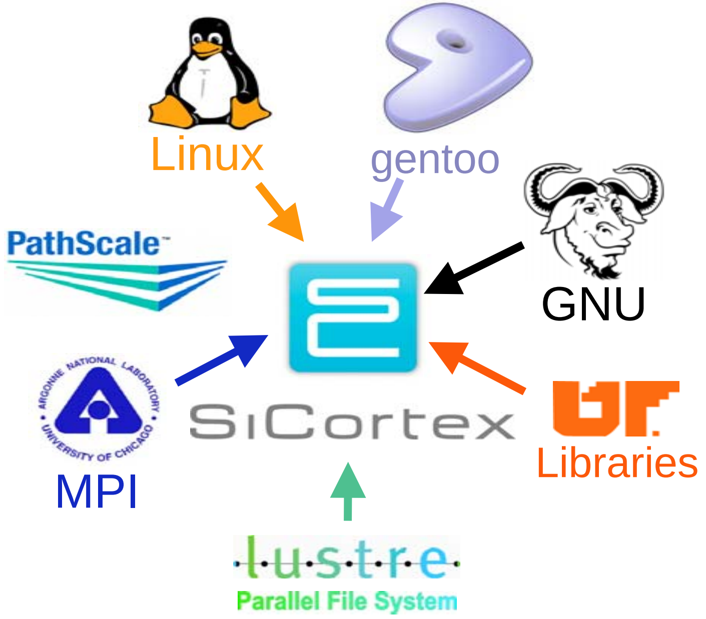
Integrated Linux/MPI Environment

Operating System
- Linux kernel and utilities
- Cluster file system (Lustre)

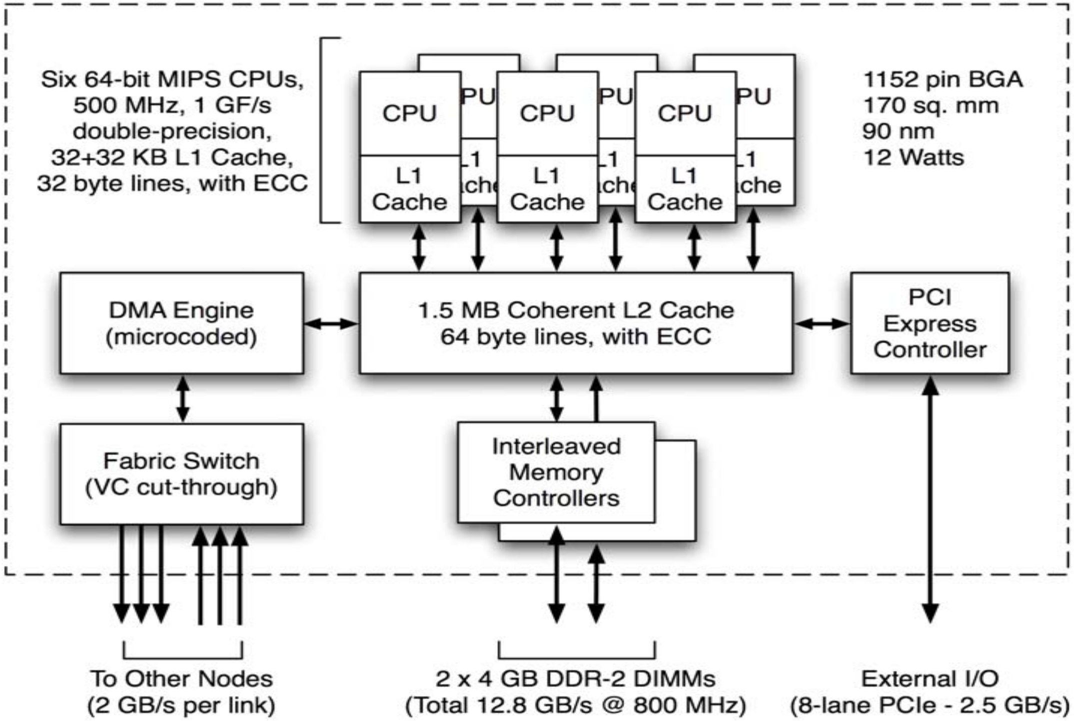
Development Environment
- GNU C, C++
- Pathscale C, C++, Fortran
- Math libraries
- Performance tools
- Debugger (TotalView)
- MPI libraries (MPICH2)

System Management
- Scheduler (SLURM + Maui)
- Partitioning
- Monitoring (Ganglia)
- Console, boot, diagnostics
- 5 Minute boot time

Maintenance and Support
- Factory-installed software
- Regular updates
- Open source build environment



Node Chip


Six 64-bit MIPS CPUs,
500 MHz, 1 GF/s
double-precision,
32+32 KB L1 Cache,
32 byte lines, with ECC
CPU
L1
Cache
1152 pin BGA
170 sq. mm
90 nm
12 Watts
DMA Engine
(microcoded)
1.5 MB Coherent L2 Cache
64 byte lines, with ECC
PCI
Express
Controller
Fabric Switch
(VC cut-through)
Interleaved
Memory
Controllers
To Other Nodes
(2 GB/s per link)
2 x 4 GB DDR-2 DIMMs
(Total 12.8 GB/s @ 800 MHz)
External I/O
(8-lane PCIe - 2.5 GB/s)

Lustre in the SiCortex system

- Application data (external storage array)
- Client for external lustre system
- Root Filesystem
- Special application data

RDMA Engine

- Implemented in hardware and microcode.
- Simple API
- Local operations, remote, chained, conditional.
- In addition to rdma, other complex operations:
  - Predefined command sequence
  - Transfer command block to remote, execute remotely
  - Conditional execution: locking, barriers etc.
- Much work still to be done to use it all.

The SiCortex Lustre Network Device

Similar to the existing ones, ideas picked up from openib lnd, ptllnd, etc.

Short immediate messages for control, small data

RDMA for large transfers

In some ways simpler than other LNDS (peer handling, no routing)

Some parts tricky due to RDMA engine characteristics.

Does a good job driving storage array via FC on IO nodes.

Ramdisk-backed Lustre

4 or 8 GB ram per node.

Create a ramdisk partition, format it as an OST.

Use the fabric-based LND to tie them together into a lustre fs.

Rootfs: ~2G total. Use a dozen or so nodes for parallelism.

FabriCache: Similar technique for application data.

Rootfs particulars

Current estimates under 2GB.

10-20 nodes, 100-200MB/node (tunable).

Readonly, to avoid some sharing issues.

Bootstrapping is a bit tricky.

Technique has been demo'ed in the lab using test hardware, seems work quite well.

FabriCache

Similar in concept to rootfs

Number of nodes based on desired size of “ramdisk”.

Rest of nodes available for computation

Alternate strategy: Use a percent of ALL nodes for ramdisk, compute on all nodes in the remaining memory.

All this tunable to fit application needs.

Current Status

The SiCortex LND is currently running on test systems in our lab.

Too early to tell about long-duration throughput, early results, look promising.

Expect to be shipping beta systems with sclnd in place, this summer.

Summary

SiCortex believes Lustre is a good solution for our customers, and does a good job of leveraging the strengths of our machine.

We expect to go to beta in early summer, with FCS late summer.

When the sclnd code stabilizes, we will be submitting it back to CFS for inclusion in the lustre 1.6.x series, going forward.